Chapter 4

State Estimation

Given a real-world system as introduced in the beginning of Chapter 2 it can not be assumed that the state $\boldsymbol{x}$ of the system is known at each time t. Nevertheless, the NMPC approach in eq. (2.20) assumes, that $\boldsymbol{x}$ is known at each discrete time t_k, for $k = 0, 1, 2, \ldots$ (for the definition of t_k see equation (2.6)). Those $n_\mathrm{y} \in \mathbb{N}_0$ process values that can be observed of a system are denoted by the measurement value function $\boldsymbol{y} : \mathbb{R}^+ \to \mathcal{Y}, \mathcal{Y} \subseteq \mathbb{R}^{n_y}$, and the functional connection of the current measurement values $\boldsymbol{y}(t)$ and the current state of the system $\boldsymbol{x}(t)$ is given by:

$$\boldsymbol{y}(t) = \boldsymbol{h}\left(\boldsymbol{x}(t), \boldsymbol{\upsilon}(t)\right). \tag{4.1}$$

Inaccuracies in the real-valued measurement function $\boldsymbol{h} : \mathcal{X} \times \mathbb{R}^{n_\upsilon} \to \mathcal{Y}$ as well as measurement noise, are modeled by the $n_\upsilon \in \mathbb{N}_0$ dimensional white Gaussian noise process $\boldsymbol{\upsilon} : \mathbb{R}^+ \to \mathbb{R}^{n_\upsilon}$ with covariance matrix $\boldsymbol{\Psi}_{\boldsymbol{\upsilon}} \in \mathbb{R}^{n_\upsilon \times n_\upsilon}$.

The question arises, whether it is possible to estimate the values of the system state $\boldsymbol{x}$ at each time t_k, given equations (2.1) and (4.1) as well as $\boldsymbol{u}(\tau)$ and $\boldsymbol{y}(\tau)$ for each $\tau \in [0, t_k]$. The state vector estimate at time t_k will be symbolized using $\hat{\boldsymbol{x}}(t_k)$ and the corresponding function is $\hat{\boldsymbol{x}} : \mathbb{R}^+ \to \mathcal{X}$. This state estimate $\hat{\boldsymbol{x}}$ will be used by the NMPC (eq. (2.20)) as an approximation of the real state $\boldsymbol{x}$ at each time t_k.

Let us assume that there are two different sampling times where the measurement $\boldsymbol{y}$ and input values $\boldsymbol{u}$ are acquired. The one for the measurement values is named $\delta_\mathrm{y} \in \mathbb{R}^+$ and the one for the input values $\delta_\mathrm{u} \in \mathbb{R}^+$. The ratio of the control sampling time δ (see Section 2.1) and both sampling times $\delta_\mathrm{y} \leq \delta$ and $\delta_\mathrm{u} \leq \delta$ are defined by the symbols:

$$N_{\delta_\mathrm{y}} := \frac{\delta}{\delta_\mathrm{y}} \in \mathbb{N}_0 \quad \text{and} \quad N_{\delta_\mathrm{u}} := \frac{\delta}{\delta_\mathrm{u}} \in \mathbb{N}_0. \tag{4.2}$$

In the following three sections (4.1 - 4.3) three different state estimation methods are proposed. Thereafter, they are applied at an anaerobic digestion process in a simulation study in Section 4.4.

4.1 State Estimation using Software Sensors

In this section an approach is developed, that tries to find a function $\boldsymbol{F}_{\mathrm{E}} : \mathcal{Y}^{N_{\delta_\mathrm{y}} \cdot k+1} \times \mathcal{U}^{N_{\delta_\mathrm{u}} \cdot k+1} \to \mathcal{X}$ with the sets

$$\mathcal{Y}^{N_{\delta_\mathrm{y}} \cdot k+1} := \{\boldsymbol{y}(0), \boldsymbol{y}(\delta_\mathrm{y}), \ldots, \boldsymbol{y}(\delta), \boldsymbol{y}(\delta+\delta_\mathrm{y}), \ldots, \boldsymbol{y}(t_k)\} \tag{4.3}$$

and

$$\mathcal{U}^{N_{\delta_\mathrm{u}} \cdot k+1} := \{\boldsymbol{u}(0), \boldsymbol{u}(\delta_\mathrm{u}), \ldots, \boldsymbol{u}(\delta), \boldsymbol{u}(\delta+\delta_\mathrm{u}), \ldots, \boldsymbol{u}(t_k)\} \tag{4.4}$$

estimating the state of the system at time t_k. This function $\boldsymbol{F}_{\mathrm{E}}$ uses the complete stream of inputs $\boldsymbol{u}$ and outputs $\boldsymbol{y}$ of the system recorded from time 0 until time t_k and therefore is a completely data based state estimator. Its returned state estimate $\hat{\boldsymbol{x}}_{\boldsymbol{F}_{\mathrm{E}}} : \mathbb{R}^+ \to \mathcal{X}$ defined as

$$\hat{\boldsymbol{x}}_{\boldsymbol{F}_{\mathrm{E}}}(t_k) := \boldsymbol{F}_{\mathrm{E}} \left(\underbrace{\boldsymbol{y}(0), \ldots, \boldsymbol{y}(t_k)}_{\in \mathcal{Y}^{N_{\delta_\mathrm{y}} \cdot k+1}}, \underbrace{\boldsymbol{u}(0), \ldots, \boldsymbol{u}(t_k)}_{\in \mathcal{U}^{N_{\delta_\mathrm{u}} \cdot k+1}} \right) \in \mathcal{X} \tag{4.5}$$

would be the best state estimate that could be achieved based on the input and output data. Unfortunately the amount of data used in this approach is increasing with time t_k, therefore in practice it will only be possible to find an approximation of this function $\boldsymbol{F}_{\mathrm{E}}$, defined as $\tilde{\boldsymbol{F}}_{\mathrm{E}} : \mathcal{Y}^{N_\mathrm{y}+1} \times \mathcal{U}^{N_\mathrm{u}+1} \to \mathcal{X}$. There a constant number of input and output samples is used, which are $N_\mathrm{u}+1 \in \mathbb{N}$ and $N_\mathrm{y}+1 \in \mathbb{N}$ using a sliding window approach. To be able to interpret N_y and N_u some formalism has to be introduced. To make the domain of $\tilde{\boldsymbol{F}}_{\mathrm{E}}$ sufficiently small, causal moving average filters are used to merge adjacent samples of input and output data to one representative value. A moving average filter for input data $\boldsymbol{\Lambda}_\mathrm{u} \in \mathcal{F}_{\boldsymbol{\Lambda}}$, with the function space of moving average filters $\mathcal{F}_{\boldsymbol{\Lambda}}$ and $\boldsymbol{\Lambda}_\mathrm{u} : \mathcal{U}^{w_\mathrm{u}} \to \mathcal{U}$, with the window size $w_\mathrm{u} \in \mathcal{W}_\mathrm{u} \subset \mathbb{N}$ is defined as:

$$\boldsymbol{\Lambda}_\mathrm{u}\left(\boldsymbol{u}(t_k), \ldots, \boldsymbol{u}\left(t_k - (w_\mathrm{u}-1)\cdot\delta_\mathrm{u}\right)\right) := \frac{1}{w_\mathrm{u}} \cdot \sum_{i=1}^{w_\mathrm{u}} \boldsymbol{u}\left(t_k - (i-1)\cdot\delta_\mathrm{u}\right). \tag{4.6}$$

Note that a moving average filter can be implemented as a tapped delay line with $w_\mathrm{u}-1$ taps.

For the input data N_u moving average filters are used, each with a different window size w_u. Thus, the set of moving average window lengths $\mathcal{W}_\mathrm{u}$ has N_u elements and is defined as $\mathcal{W}_\mathrm{u} := \{w_{\mathrm{u},1}, \ldots, w_{\mathrm{u},N_\mathrm{u}}\}$. Then, to each window size $w_{\mathrm{u},i_{\boldsymbol{\Lambda}_\mathrm{u}}}$ belongs the moving average filter $\boldsymbol{\Lambda}_{\mathrm{u},i_{\boldsymbol{\Lambda}_\mathrm{u}}} \in \mathcal{F}_{\boldsymbol{\Lambda}}$, returning for each $i_{\boldsymbol{\Lambda}_\mathrm{u}} = 1, \ldots, N_\mathrm{u}$ the moving average value $\overline{\boldsymbol{u}}_{i_{\boldsymbol{\Lambda}_\mathrm{u}}} : \mathbb{R}^+ \to \mathcal{U}$ defined as:

$$\overline{\boldsymbol{u}}_{i_{\boldsymbol{\Lambda}_\mathrm{u}}}(t_k) := \boldsymbol{\Lambda}_{\mathrm{u},i_{\boldsymbol{\Lambda}_\mathrm{u}}}\left(\boldsymbol{u}(t_k), \ldots, \boldsymbol{u}\left(t_k - \left(w_{\mathrm{u},i_{\boldsymbol{\Lambda}_\mathrm{u}}} - 1\right)\cdot\delta_\mathrm{u}\right)\right) \in \mathcal{U}. \tag{4.7}$$

We equally define a moving average filter for output data $\boldsymbol{\Lambda}_\mathrm{y} \in \mathcal{F}_{\boldsymbol{\Lambda}}$, $\boldsymbol{\Lambda}_\mathrm{y} : \mathcal{Y}^{w_\mathrm{y}} \to \mathcal{Y}$,

with the window size $w_\mathrm{y} \in \mathcal{W}_\mathrm{y} \subset \mathbb{N}_0$ as

$$\boldsymbol{\Lambda}_\mathrm{y}\left(\boldsymbol{y}(t_k), \ldots, \boldsymbol{y}\left(t_k - (w_\mathrm{y}-1)\cdot\delta_\mathrm{y}\right)\right) := \frac{1}{w_\mathrm{y}} \cdot \sum_{i=1}^{w_\mathrm{y}} \boldsymbol{y}\left(t_k - (i-1)\cdot\delta_\mathrm{y}\right). \tag{4.8}$$

For the measurement data N_y moving average filters are used, each with a different window size w_y. Thus, the set of moving average window lengths $\mathcal{W}_\mathrm{y}$ has N_y elements and is defined as $\mathcal{W}_\mathrm{y} := \left\{w_{\mathrm{y},1}, \ldots, w_{\mathrm{y},N_\mathrm{y}}\right\}$. Then, to each window size $w_{\mathrm{y},i_{\boldsymbol{\Lambda}_\mathrm{y}}}$ the moving average filter $\boldsymbol{\Lambda}_{\mathrm{y},i_{\boldsymbol{\Lambda}_\mathrm{y}}} \in \mathcal{F}_{\boldsymbol{\Lambda}}$ belongs, returning for each $i_{\boldsymbol{\Lambda}_\mathrm{y}} = 1, \ldots, N_\mathrm{y}$ the moving average value $\overline{\boldsymbol{y}}_{i_{\boldsymbol{\Lambda}_\mathrm{y}}} : \mathbb{R}^+ \to \mathcal{Y}$ defined as:

$$\overline{\boldsymbol{y}}_{i_{\boldsymbol{\Lambda}_\mathrm{y}}}(t_k) := \boldsymbol{\Lambda}_{\mathrm{y},i_{\boldsymbol{\Lambda}_\mathrm{y}}}\left(\boldsymbol{y}(t_k), \ldots, \boldsymbol{y}\left(t_k - \left(w_{\mathrm{y},i_{\boldsymbol{\Lambda}_\mathrm{y}}} - 1\right)\cdot\delta_\mathrm{y}\right)\right) \in \mathcal{Y}. \tag{4.9}$$

The vector which is returned by the approximate state estimation function $\tilde{\boldsymbol{F}}_\mathrm{E}$, defined above, is used as state estimate at each time t_k:

$$\hat{\boldsymbol{x}}(t_k) := \tilde{\boldsymbol{F}}_\mathrm{E}\left(\underbrace{\boldsymbol{y}(t_k), \overline{\boldsymbol{y}}_1(t_k), \ldots, \overline{\boldsymbol{y}}_{N_\mathrm{y}}(t_k)}_{\in\, \mathcal{Y}^{N_\mathrm{y}+1}}, \underbrace{\boldsymbol{u}(t_k), \overline{\boldsymbol{u}}_1(t_k), \ldots, \overline{\boldsymbol{u}}_{N_\mathrm{u}}(t_k)}_{\in\, \mathcal{U}^{N_\mathrm{u}+1}}\right). \tag{4.10}$$

Now it is established which values are passed to the estimation function $\tilde{\boldsymbol{F}}_\mathrm{E}$. But, what kind of function $\tilde{\boldsymbol{F}}_\mathrm{E}$ should be is not yet clear. At the moment, and there maybe never will be, an equation which describes for a biogas plant how current and past input and output values let one imply on the current state of the system. Therefore, this function has to be found. In this thesis it was tried to find an approximation for this function using supervised machine learning methods. In supervised learning many matching input and output data samples are presented to the learning method. The method tries to find a general structure in the seen data, which is then generalized to all unseen regions of the data space. In the following subsection this idea is made more clear. Furthermore, three machine learning methods are briefly introduced, which were used to find a model for the state estimation function $\tilde{\boldsymbol{F}}_\mathrm{E}$. These are Random Forests, linear discriminant analysis and generalized discriminant analysis, whereas both discriminant analysis methods are used as data preprocessing methods for a linear classifier. As machine learning methods in general can be highly nonlinear, the state estimator $\tilde{\boldsymbol{F}}_\mathrm{E}$ can be highly nonlinear as well. In contrast to a dynamic state estimation method this one is static. Therefore, issues such as stability and convergence are not applicable. The estimator is always stable. Its estimate does not converge to the real state values, but there remains a data dependent estimation error.

4.1.1 Supervised Machine Learning Methods

To be able to apply machine learning methods training and validation samples are created as follows. Without loss of generalization let us set $\delta_\mathrm{u} = \delta_\mathrm{y}$. Then the matrices

$$\boldsymbol{Y} := \begin{pmatrix} \boldsymbol{y}^T(0), \overline{\boldsymbol{y}}_1^T(0), \ldots, \overline{\boldsymbol{y}}_{N_\mathrm{y}}^T(0), \boldsymbol{u}^T(0), \overline{\boldsymbol{u}}_1^T(0), \ldots, \overline{\boldsymbol{u}}_{N_\mathrm{u}}^T(0) \\ \boldsymbol{y}^T\left(\delta_\mathrm{y}\right), \overline{\boldsymbol{y}}_1^T\left(\delta_\mathrm{y}\right), \ldots, \overline{\boldsymbol{y}}_{N_\mathrm{y}}^T\left(\delta_\mathrm{y}\right), \boldsymbol{u}^T\left(\delta_\mathrm{y}\right), \overline{\boldsymbol{u}}_1^T\left(\delta_\mathrm{y}\right), \ldots, \overline{\boldsymbol{u}}_{N_\mathrm{u}}^T\left(\delta_\mathrm{y}\right) \\ \vdots \\ \boldsymbol{y}^T(t_k), \overline{\boldsymbol{y}}_1^T(t_k), \ldots, \overline{\boldsymbol{y}}_{N_\mathrm{y}}^T(t_k), \boldsymbol{u}^T(t_k), \overline{\boldsymbol{u}}_1^T(t_k), \ldots, \overline{\boldsymbol{u}}_{N_\mathrm{u}}^T(t_k) \end{pmatrix}, \tag{4.11}$$

$\boldsymbol{Y} \in \mathbb{R}^{N \times D}$, $D := n_\mathrm{y} \cdot (N_\mathrm{y} + 1) + n_\mathrm{u} \cdot (N_\mathrm{u} + 1)$, $N := k \cdot N_{\delta_\mathrm{y}} + 1$, and

$$\boldsymbol{X} := \left(\boldsymbol{x}_{i_\mathrm{x}}, \ldots, \boldsymbol{x}_{n_\mathrm{x}}\right) := \begin{pmatrix} \boldsymbol{x}^T(0) \\ \boldsymbol{x}^T\left(\delta_\mathrm{y}\right) \\ \vdots \\ \boldsymbol{x}^T(t_k) \end{pmatrix} \in \mathbb{R}^{N \times n_\mathrm{x}} \tag{4.12}$$

can be defined, with $\boldsymbol{x}_{i_\mathrm{x}} \in \mathbb{R}^N$. Using both matrices $\boldsymbol{X}$ and $\boldsymbol{Y}$, the state estimation problem is to find a mapping $\boldsymbol{Y} \mapsto \boldsymbol{x}_{i_\mathrm{x}}$ for each state vector component $i_\mathrm{x} = 1, \ldots, n_\mathrm{x}$. As said in the beginning of this chapter it cannot be assumed that the state $\boldsymbol{x}$ is available at each discrete time t_k. Therefore, the matrix $\boldsymbol{X}$ is not available. Hence, a calibrated simulation model of the biogas plant at hand is used to generate an approximation of $\boldsymbol{X}$, replacing $\boldsymbol{x}$ with ${}^o\boldsymbol{x}$ at each simulated time τ. The simulation model consists out of eqs. (2.1) and (4.1). At the same time all vectors $\boldsymbol{y}$ in $\boldsymbol{Y}$ are replaced with the values returned by $\boldsymbol{h}\left({}^o\boldsymbol{x}(\tau), \boldsymbol{v}(\tau)\right)$ at each corresponding time τ. Based on ${}^o\boldsymbol{x}$ and $\boldsymbol{h}$, an approximation of the original problem is solved, assuming that the model emulates the real process with sufficient accuracy.

This estimation problem can be solved using either regression or classification techniques. In this case, classification was used.

To be able to apply discriminant analysis and classification methods on the dataset, the range for each state vector component $\boldsymbol{x}_{i_\mathrm{x}}$ is clustered into $C \in \mathbb{N}_0$ equally distributed classes, $i_\mathrm{x} = 1, \ldots, n_\mathrm{x}$. Thus, vectors are generated containing the class labels corresponding to the simulated values of the state vector components $\boldsymbol{x}_{i_\mathrm{x}}$, that is, $\boldsymbol{\vartheta}_{i_\mathrm{x}} \in \{1, ..., C\}^N$, $i_\mathrm{x} = 1, \ldots, n_\mathrm{x}$.

Before machine learning methods are applied, the complete dataset $\boldsymbol{Y}$ is split into a training dataset $\boldsymbol{Y}_\mathrm{T} \in \mathbb{R}^{N_\mathrm{T} \times D}$ and a validation dataset $\boldsymbol{Y}_\mathrm{V} \in \mathbb{R}^{N_\mathrm{V} \times D}$ with $N_\mathrm{V} := N - N_\mathrm{T}$, $N_\mathrm{T} < N$. In the following, the used machine learning methods are briefly described.

4.1.1.1 Linear Discriminant Analysis (LDA)

Linear discriminant analysis searches for a linear transformation $\boldsymbol{A}_{\mathrm{LDA}} \in \mathbb{R}^{d \times D}$, $d \leq D$, such that the transformed data $\boldsymbol{Z} = \boldsymbol{A}_{\mathrm{LDA}} \cdot \boldsymbol{Y}_{\mathrm{T}}^{T}$, $\boldsymbol{Z} := (\boldsymbol{z}_1, \ldots, \boldsymbol{z}_{N_{\mathrm{T}}}) \in \mathbb{R}^{d \times N_{\mathrm{T}}}$, can be linearly separated better than the original feature vectors $\boldsymbol{Y}_{\mathrm{T}}^{T}$. The linear transformation $\boldsymbol{A}_{\mathrm{LDA}}$ is determined by solving an optimization problem maximizing the well-known Fisher discriminant criterion:

$$\operatorname{trace}\left\{\boldsymbol{S}_{\mathrm{T}}^{-1} \cdot \boldsymbol{S}_{\mathrm{B}}\right\} \tag{4.13}$$

where $\boldsymbol{S}_{\mathrm{T}} \in \mathbb{R}^{D \times D}$ is total scatter-matrix and $\boldsymbol{S}_{\mathrm{B}} \in \mathbb{R}^{D \times D}$ is the between-class scatter-matrix for the data (Duda et al., 2000). The LDA and a subsequent linear classifier are both implemented in MATLAB®.

4.1.1.2 Generalized Discriminant Analysis (GerDA) (Stuhlsatz et al., 2012)

LDA is a popular pre-processing and visualization tool used in different pattern recognition applications. Unfortunately, LDA followed by linear classification produces high error rates on many real-world datasets, because a linear mapping $\boldsymbol{A}_{\mathrm{LDA}}$ cannot transform arbitrarily distributed features into independently Gaussian distributed ones. A natural generalization of the classical LDA is to assume a function space $\mathcal{F}$ of nonlinear transformations $\boldsymbol{f}_{\mathrm{GerDA}} : \mathbb{R}^{D} \rightarrow \mathbb{R}^{d}$ and to still rely on having intrinsic features $\boldsymbol{z}_i := \boldsymbol{f}_{\mathrm{GerDA}}(\boldsymbol{y}_i)$, $i = 1, \ldots, N_{\mathrm{T}}$, with the same statistical properties as assumed for LDA features. The idea is that a sufficiently large space $\mathcal{F}$ potentially contains a nonlinear feature extractor $\boldsymbol{f}_{\mathrm{GerDA}}^{*} \in \mathcal{F}$ that increases the discriminant criterion (4.13) compared with a linear extractor $\boldsymbol{A}_{\mathrm{LDA}}$. GerDA defines a large space $\mathcal{F}$ using a deep neural network (DNN), and consequently the nonlinear feature extractor $\boldsymbol{f}_{\mathrm{GerDA}}^{*} \in \mathcal{F}$ is given by the DNN which is trained with measurements of the data space such that the objective function (4.13) is maximized. Unfortunately, training a DNN with standard methods, like back-propagation, is known to be challenging due to many local optima in the considered objective function. To efficiently train a large DNN with respect to (4.13), in Stuhlsatz et al. (2010a,b) a stochastic pre-optimization has been proposed based on greedily layer-wise trained Restricted Boltzmann Machines (Hinton et al., 2006). After layer-wise pre-optimization all weights $\boldsymbol{W}$ and biases $\boldsymbol{b}$ of the GerDA-DNN are appropriately initialized. Nevertheless, pre-optimization is suboptimal in maximizing (4.13), thus a subsequent fine-tuning of the GerDA-DNN is performed using a modified back-propagation of the gradients of (4.13) with respect to the network parameters. In Stuhlsatz et al. (2010a,b) it is shown that stochastic pre-optimization and subsequent fine-tuning yields very good discriminative features and training time is substantially reduced compared with random initialization of large GerDA-DNNs. The GerDA-framework is implemented in MATLAB®.

4.1.1.3 Random Forests

Random Forests consists out of an ensemble of decision trees (Breiman, 2001) and can be used to solve complex classification and regression problems. At each node of such a binary decision tree the dataset at that node is split into two disjoint datasets. At each leaf of the tree the value for the predicted variable is decided. Classification is performed by taking the majority vote of an ensemble of classification trees, where each tree is trained on a bootstrapped sample of the original training dataset. This results in an ensemble of slightly different decision trees leading to improved generalization (Criminisi et al., 2011). The Random Forests algorithm used is from the Random Forests implementation for MATLAB® (and Standalone) (Jaiantilal, 2010).

4.2 Hybrid Extended Kalman Filter

Above eq. (4.2) the sampling time for measurement values δ_{y} was introduced. The hybrid extended Kalman filter proposed in this section will return a state estimate at each time

$$t_j := j \cdot \delta_{\mathrm{y}} \tag{4.14}$$

for $j = 1, 2, 3, \ldots$. Setting $t_j = t_k$ with t_k defined in equation (2.6) and using eq. (4.2) it is

$$j = N_{\delta_{\mathrm{y}}} \cdot k. \tag{4.15}$$

Thus, j runs with $N_{\delta_{\mathrm{y}}}$ times the frequency of k. At time instant $j = 0$ the filter is started and initialized with the expectation value of the system state at time instant $k = 0$: $E\left\langle \boldsymbol{x}\left(t_0\right)\right\rangle$.
To simplify the notation it is generally defined:

$$\boldsymbol{X}_j := \boldsymbol{X}(t_j) \quad \text{and} \quad \boldsymbol{x}_j := \boldsymbol{x}(t_j) \tag{4.16}$$

as well as

$$\boldsymbol{X}_k := \boldsymbol{X}(t_k) \quad \text{and} \quad \boldsymbol{x}_k := \boldsymbol{x}(t_k) \tag{4.17}$$

for any matrix $\boldsymbol{X}(t_j), \boldsymbol{X}(t_k) \in \mathbb{R}^{m \times n}$ and any vector $\boldsymbol{x}(t_j), \boldsymbol{x}(t_k) \in \mathbb{R}^n$, $n, m \in \mathbb{N}$.
A Kalman filter basically can be divided into the two parts prediction and correction. In the prediction step the model equation of the system (eq. (2.1)) is used to predict the current state $\boldsymbol{x}_j$ of the system given a state estimate from the last iteration $j-1$. In the correction step the current measurement values $\boldsymbol{y}_j$ are taken to correct the predicted state estimate. The state estimate at time instant j is named the a priori state estimate, denoted by $\hat{\boldsymbol{x}}_j^- := \hat{\boldsymbol{x}}\left(t_j^-\right) \in \mathcal{X}$, and the corrected state estimate is the a posteriori state estimate $\hat{\boldsymbol{x}}_j^+ := \hat{\boldsymbol{x}}\left(t_j^+\right) \in \mathcal{X}$. In Figure 4.1 the idea of both definitions and the meaning of the times t_j^- and t_j^+ are visualized, $t_j^- \lessapprox t_j \lessapprox t_j^+$.

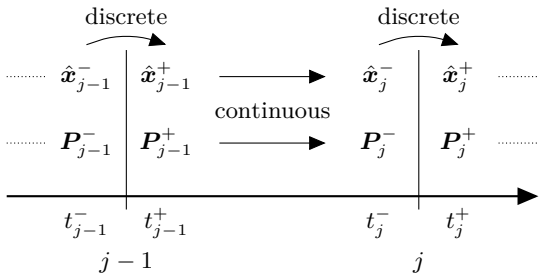

Figure 4.1: Definition of a priori ($\hat{\boldsymbol{x}}_j^-$, $\boldsymbol{P}_j^-$) and a posteriori state estimates and estimation error covariance matrices ($\hat{\boldsymbol{x}}_j^+$, $\boldsymbol{P}_j^+$), respectively (cf. Simon (2006)).

The propagation of the estimation error covariance matrix

$$\boldsymbol{P}_j := E\left\langle (\boldsymbol{x}_j - \hat{\boldsymbol{x}}_j) \cdot (\boldsymbol{x}_j - \hat{\boldsymbol{x}}_j)^T \right\rangle \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}} \tag{4.18}$$

is visualized in Figure 4.1 as well. The a priori $\boldsymbol{P}_j^- := \boldsymbol{P}\left(t_j^-\right) \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$ and a posteriori estimation error covariance matrices $\boldsymbol{P}_j^+ := \boldsymbol{P}\left(t_j^+\right) \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{x}}$ describe the certainty in the corresponding state estimate at each time t_j^- and t_j^+, respectively. In the hybrid extended Kalman filter the prediction step is done in continuous-time and the correction step is calculated in discrete time. This filter is dedicated to nonlinear systems that are continuous in nature, but where the measurements $\boldsymbol{y}$ are measured discretely with a sampling time δ_y.
The algorithm of the hybrid extended Kalman filter can be described as follows (Simon, 2006).

1. The system equations with continuous-time dynamics and discrete-time measurements are given as follows (with the Kronecker delta $\delta_{j_1 j_2}$), (Grewal and Andrews, 2008):

   eq. (2.1) $\quad {}^o\boldsymbol{x}'(t) = \boldsymbol{f}\left({}^o\boldsymbol{x}(t), \boldsymbol{u}(t), \boldsymbol{\omega}(t)\right)$

   eq. (4.1) $\quad \boldsymbol{y}(t_j) = \boldsymbol{h}\left(\boldsymbol{x}(t_j), \boldsymbol{v}_j\right)$

   $$\boldsymbol{\omega}(t) \sim \mathcal{N}\left(\boldsymbol{0}, \boldsymbol{\Psi}_{\boldsymbol{\omega}}\right)$$

   $$\boldsymbol{v}_j \sim \mathcal{N}\left(\boldsymbol{0}, \frac{\boldsymbol{\Psi}_{\boldsymbol{v}}}{\delta_\mathrm{y}}\right) \qquad E\left\langle \boldsymbol{v}_{j_1} \cdot \boldsymbol{v}_{j_2}^T \right\rangle = \delta_{j_1 j_2} \cdot \frac{\boldsymbol{\Psi}_{\boldsymbol{v}}}{\delta_\mathrm{y}}$$

2. Initialize the filter as follows:

   $$\hat{\boldsymbol{x}}_0^+ = E\left\langle \boldsymbol{x}_0 \right\rangle \qquad \boldsymbol{P}_0^+ = E\left\langle \left(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0^+\right)\left(\boldsymbol{x}_0 - \hat{\boldsymbol{x}}_0^+\right)^T \right\rangle$$

3. For $j = 1, 2, \ldots$ perform the following:

   (a) Integrate the state estimate and its covariance from time t_{j-1}^+ to time t_j^- as

follows:

$$
\begin{aligned}
{}^{o}\boldsymbol{x}'(\tau) &= \boldsymbol{f}\left({}^{o}\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{0}\right) \\
\boldsymbol{P}'(\tau) &= \boldsymbol{A}_j \cdot \boldsymbol{P}(\tau) + \boldsymbol{P}(\tau) \cdot \boldsymbol{A}_j^T + \boldsymbol{E}_j \cdot \boldsymbol{\Psi}_{\boldsymbol{\omega}} \cdot \boldsymbol{E}_j^T \\
\tau \in \left[t_{j-1}^+, t_j^-\right] & \qquad {}^{o}\boldsymbol{x}\left(t_{j-1}^+\right) = \hat{\boldsymbol{x}}_{j-1}^+
\end{aligned}
$$

with the linearizations:

$$
\begin{aligned}
\boldsymbol{A}_j &:= \frac{\partial \boldsymbol{f}}{\partial {}^{o}\boldsymbol{x}}\left(\hat{\boldsymbol{x}}_{j-1}^+, \boldsymbol{u}(t_j), \boldsymbol{0}\right) & \boldsymbol{C}_j &:= \frac{\partial \boldsymbol{h}}{\partial {}^{o}\boldsymbol{x}}\left(\hat{\boldsymbol{x}}_{j-1}^+, \boldsymbol{0}\right) \\
\boldsymbol{E}_j &:= \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{\omega}}\left(\hat{\boldsymbol{x}}_{j-1}^+, \boldsymbol{u}(t_j), \boldsymbol{0}\right) & \boldsymbol{F}_j &:= \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{\upsilon}}\left(\hat{\boldsymbol{x}}_{j-1}^+, \boldsymbol{0}\right)
\end{aligned}
\tag{4.19}
$$

At the end of this integration we set $\hat{\boldsymbol{x}}_j^- = {}^{o}\boldsymbol{x}\left(t_j^-\right)$.

(b) At time instant j, incorporate the measurement $\boldsymbol{y}_j$ into the state estimate and estimation covariance as follows ($\boldsymbol{K}_j^* \in \mathbb{R}^{n_\mathrm{x} \times n_\mathrm{y}}$ is called optimal Kalman matrix):

$$
\begin{aligned}
\boldsymbol{K}_j^* &= \boldsymbol{P}_j^- \cdot \boldsymbol{C}_j^T \cdot \left(\boldsymbol{C}_j \cdot \boldsymbol{P}_j^- \cdot \boldsymbol{C}_j^T + \boldsymbol{F}_j \cdot \boldsymbol{\Psi}_{\boldsymbol{\upsilon}} \cdot \boldsymbol{F}_j^T\right)^{-1} \\
\hat{\boldsymbol{x}}_j^+ &= \hat{\boldsymbol{x}}_j^- + \boldsymbol{K}_j^* \cdot \left(\boldsymbol{y}_j - \boldsymbol{h}\left(\hat{\boldsymbol{x}}_j^-, \boldsymbol{0}\right)\right) \\
\boldsymbol{P}_j^+ &= \left(\boldsymbol{1}_{n_\mathrm{x}} - \boldsymbol{K}_j^* \cdot \boldsymbol{C}_j\right) \cdot \boldsymbol{P}_j^-
\end{aligned}
\tag{4.20}
$$

The last equation in eq. (4.20) can be replaced by the equivalent expression

$$
\boldsymbol{P}_j^+ = \left(\boldsymbol{1}_{n_\mathrm{x}} - \boldsymbol{K}_j^* \cdot \boldsymbol{C}_j\right) \boldsymbol{P}_j^- \left(\boldsymbol{1}_{n_\mathrm{x}} - \boldsymbol{K}_j^* \cdot \boldsymbol{C}_j\right)^T + \boldsymbol{K}_j^* \cdot \boldsymbol{\Psi}_{\boldsymbol{\upsilon}} \cdot \left(\boldsymbol{K}_j^*\right)^T
$$

which can be shown to be more robust (Simon, 2006).

4.3 Moving Horizon Estimation

Moving horizon state estimation (MHE) estimates the current state $\boldsymbol{x}_k$ out of input and output measurements coming from the system, starting in the past and reaching into present. Therefore, simulations with the plant model (eqs. (2.1) and (4.1)) are started in the past from different initial states $\boldsymbol{x}_\circ \in \mathcal{X}$ using the given input data $\boldsymbol{u}$. The obtained simulation results are compared with the output measurements $\boldsymbol{y}$, and the initial state leading to the best agreement of both trajectories (named ${\boldsymbol{x}_\circ}^* \in \mathcal{X}$) is used to generate the trajectory of states eventually leading to the current state estimate $\hat{\boldsymbol{x}}(t_k)$. Using the moving window concept, this approach is repeated at every sampling instance k of the control. To put this idea into formalism, the length of the moving horizon $\delta_\mathrm{MHE} \in \mathbb{R}^+$ is defined as

$$
\delta_\mathrm{MHE} := w_\mathrm{MHE} \cdot \delta_\mathrm{y} \tag{4.21}
$$

with the unit-less length of the horizon $w_\mathrm{MHE} \in \mathbb{N}$ and the sampling time of the measurement values δ_y, see eq. (4.2). For later use another unit-less length of the

horizon $\tilde{w}_{\mathrm{MHE}} \in \mathbb{N}$ is defined as:

$$\tilde{w}_{\mathrm{MHE}} := \frac{1}{N_{\delta_\mathrm{y}}} \cdot w_{\mathrm{MHE}} \overset{(4.2)}{=} \frac{\delta_\mathrm{y}}{\delta} \cdot w_{\mathrm{MHE}} \overset{(4.21)}{=} \frac{\delta_{\mathrm{MHE}}}{\delta}. \tag{4.22}$$

From eq. (4.22) it can be seen that the following relation holds:

$$\frac{\tilde{w}_{\mathrm{MHE}}}{w_{\mathrm{MHE}}} = \frac{\delta_\mathrm{y}}{\delta},$$

thus $\tilde{w}_{\mathrm{MHE}}$ measures the window-length in the units of δ and w_{MHE} the same in the units of δ_y. Keeping in mind the definitions of t_k and t_j in eqs. (2.6) and (4.14), respectively, and using the system equations (2.1) and (4.1) the moving horizon state estimation problem can be formulated as follows (cf. Busch et al. (2009)):

For each $k = \tilde{w}_{\mathrm{MHE}}, \tilde{w}_{\mathrm{MHE}} + 1, \tilde{w}_{\mathrm{MHE}} + 2, \ldots$ solve:

$$\begin{aligned}
\boldsymbol{x}_\circ{}^* := \arg\min_{\boldsymbol{x}_\circ} \Bigg[& \left\| \boldsymbol{x}_\circ - \tilde{\boldsymbol{x}}\left(t_k - \delta_{\mathrm{MHE}}\right) \right\|_2 \; + \\
& + \kappa_{\mathrm{MHE}} \cdot \sum_{j=N_{\delta_\mathrm{y}} \cdot (k - \tilde{w}_{\mathrm{MHE}})}^{N_{\delta_\mathrm{y}} \cdot k} \left\| \boldsymbol{y}\left(t_j\right) - \boldsymbol{h}\left({}^o\boldsymbol{x}\left(t_j\right), \boldsymbol{v}\left(t_j\right)\right) \right\|_2 \Bigg] \\
\text{subject to} \quad & {}^o\boldsymbol{x}'(\tau) = \boldsymbol{f}\left({}^o\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{0}\right), && {}^o\boldsymbol{x}\left(t_k - \delta_{\mathrm{MHE}}\right) = \boldsymbol{x}_\circ, \\
& {}^o\boldsymbol{x}(\tau) \in \mathcal{X}, \boldsymbol{u}(\tau) \in \mathcal{U}, && \forall \tau \in \left[t_k - \delta_{\mathrm{MHE}}, t_k\right], \\
& \boldsymbol{x}_{\mathrm{LB}} \leq \boldsymbol{x}_\circ \leq \boldsymbol{x}_{\mathrm{UB}}.
\end{aligned} \tag{4.23}$$

The state estimate at time t_k, $\hat{\boldsymbol{x}}(t_k)$, then is given by the final value of the simulation starting at the optimal initial state $\boldsymbol{x}_\circ{}^*$:

$$\begin{aligned}
& \hat{\boldsymbol{x}}(t_k) = {}^o\boldsymbol{x}(t_k) && \tilde{\boldsymbol{x}}_\circ := {}^o\boldsymbol{x}\left(t_{k+1} - \delta_{\mathrm{MHE}}\right) \in \mathcal{X} \\
& {}^o\boldsymbol{x}'(\tau) = \boldsymbol{f}\left({}^o\boldsymbol{x}(\tau), \boldsymbol{u}(\tau), \boldsymbol{0}\right) && {}^o\boldsymbol{x}\left(t_k - \delta_{\mathrm{MHE}}\right) = \boldsymbol{x}_\circ{}^* \qquad \forall \tau \in \left[t_k - \delta_{\mathrm{MHE}}, t_k\right]
\end{aligned}$$

In eq. (4.23) $\tilde{\boldsymbol{x}} : \mathbb{R}^+ \to \mathcal{X}$ denotes the initial state estimate at the start of the moving window, $\kappa_{\mathrm{MHE}} \in \mathbb{R}^+$ denotes a weighting factor and $\boldsymbol{x}_{\mathrm{LB}}, \boldsymbol{x}_{\mathrm{UB}} \in \mathcal{X}$. At the first start of the estimator an initial estimate of $\tilde{\boldsymbol{x}}\left(0\right)$ must be given, all later iterations can generate the initial estimate from the previous optimal simulation. This so called arrival cost can be computed by Kalman filter updates (Busch et al., 2009). Although Diehl et al. (2006b) strictly advises against the very simplified approach of setting $\tilde{\boldsymbol{x}}\left(t_k - \delta_{\mathrm{MHE}}\right) = \tilde{\boldsymbol{x}}_\circ$ for each $k = \tilde{w}_{\mathrm{MHE}} + 1, \tilde{w}_{\mathrm{MHE}} + 2, \ldots$, this approach is used in the small application given in Section 4.4 to keep things as simple as possible. But, for real systems more sophisticated approaches such as the one in Diehl et al. (2006b) should be used instead. It should be mentioned that the moving horizon state estimation problem can be extended easily with parameter estimation, see Busch et al. (2009).

As the optimization problem stated in eq. (4.23) must be solved for every sampling time t_k, real-time approaches are an important issue. For large-scale and fast systems more sophisticated approaches to solve the optimization problem, especially differential equation (2.1), must be used. Such methods are direct multiple shooting and its real-time implementation (Diehl et al., 2006b,c) as well as approximate solutions (Alessandri et al., 2008, 2010, 2011). Using one of these approaches, very large systems can be solved in real-time, e.g. Busch et al. (2009), Diehl et al. (2006c).

4.4 Application to an Anaerobic Digestion Process

In this section the three proposed state estimation methods are applied to a simple anaerobic digestion model and their performances are compared in a simulation study. The used model was developed by Marsili-Libelli and Beni (1996) and adapted by Shen et al. (2006). Here the implementation and parametrization of the latter is used. An introduction into the anaerobic digestion process is given in Chapter 5. In the following paragraph the applied simulation model is introduced briefly.

The model is a two-population model representing two species of bacteria: acidogenic bacteria (acidogens X_a) and methanogenic bacteria (methanogens X_m). The acidogenic bacteria convert the organic substrate S into acetic acid V_a and carbon dioxide C and the methanogenic bacteria convert the acetic acid V_a into methane Q_{ch_4} and carbon dioxide C as well. The produced gas is transferred between the liquid and gas phase resulting in biogas production Q_{ch_4} and Q_{co_2}. Furthermore, the association and dissociation of acetic acid and sodium bicarbonate as well as the effects of CO_2 and bicarbonate on the liquid phase pH are modelled (Shen et al., 2006). This results in a model containing six ordinary differential equations and two independent inputs S and B_{ic}. Its important variables are given in Table 4.1. All equations and parameters of the model can be found in the appendix of this thesis, Part A. To be able to formulate the

Table 4.1: The most important model variables as in Shen et al. (2006)

Variable	Unit	Description
S	[mg/l]	Organic substrate concentration
X_a	[mg/l]	Acidogenic bacteria concentration
V_a	[mg/l]	Acetic acid concentration
X_m	[mg/l]	Methanogenic bacteria concentration
C	[mg/l]	Carbon dioxide concentration (liquid phase)
P_c	[mg/l]	Carbon dioxide partial pressure (gas phase)
Q_{ch_4}	[l/h]	Methane gas production
Q_{co_2}	[l/h]	Carbon dioxide gas production
S_i	[mg/l]	Influent organic substrate concentration
B_{ic}	[mg/l]	Cation ions concentration introduced by sodium bicarbonate

given model in the standard notation given by equations (2.1) and (4.1), it is defined:

$$\begin{aligned} {}^{o}\boldsymbol{x} &:= (S, X_a, V_a, X_m, C, P_c)^T & n_{\mathrm{x}} &= 6 \\ \boldsymbol{u} &:= (S_i, B_{ic})^T & n_{\mathrm{u}} &= 2 \\ \boldsymbol{y} &:= (Q_{\mathrm{ch_4}}, Q_{\mathrm{co_2}})^T & n_{\mathrm{y}} &= 2. \end{aligned} \tag{4.24}$$

Note, that methane and carbon dioxide production $Q_{\mathrm{ch_4}}$ and $Q_{\mathrm{co_2}}$ can be easily measured, so that the such defined measurement vector $\boldsymbol{y}$ can be determined in practice without extraordinary effort. In the following, results for experiments performed for the three state estimation methods are presented.

4.4.1 The Experiments

To compare the three different state estimation methods the following setup is chosen. 30 different simulations from ten different randomly selected initial states with three different input trajectories are performed. The three different input trajectories $\boldsymbol{u}_\alpha := (S_{i,\alpha}, B_{ic,\alpha})^T$, $\alpha = 1, 2, 3$, are visualized in Figure 4.2. The first ten simulations are performed with input $\boldsymbol{u}_1$, the next 10 simulations with $\boldsymbol{u}_2$ and then the last ten simulations with the input trajectory $\boldsymbol{u}_3$. The simulation duration for each one is set to 100 days.

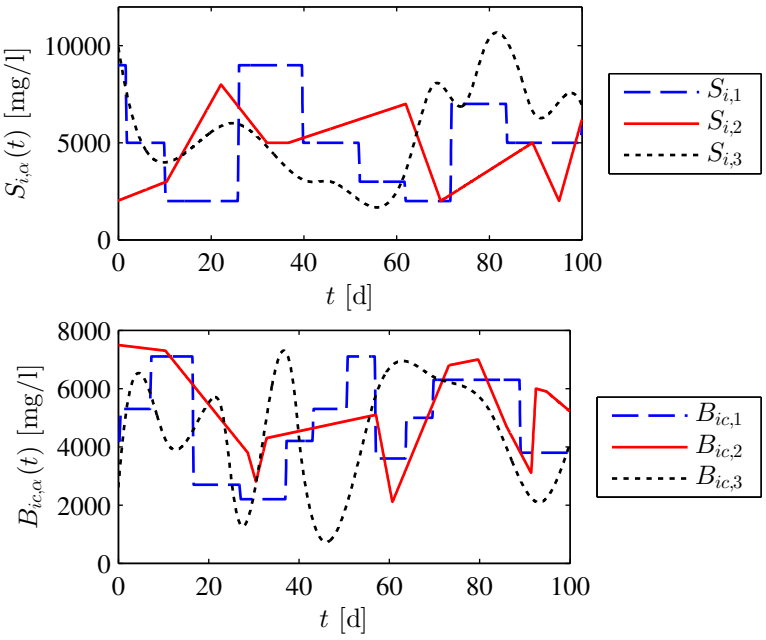

Figure 4.2: Input trajectories $\boldsymbol{u}_\alpha := (S_{i,\alpha}, B_{ic,\alpha})^T$, $\alpha = 1, 2, 3$

The noise processes $\boldsymbol{\omega}$ and $\boldsymbol{\upsilon}$ are solely modelled as additive input and measurement noise, respectively. Therefore, from eqs. (2.1) and (4.1) it follows:

$$\begin{aligned} {}^{o}\boldsymbol{x}'(t) &= \boldsymbol{f}\left({}^{o}\boldsymbol{x}(t), \boldsymbol{u}(t) + \boldsymbol{\omega}(t), \boldsymbol{0}\right) \\ \boldsymbol{y}(t) &= \boldsymbol{h}\left(\boldsymbol{x}(t), \boldsymbol{0}\right) + \boldsymbol{\upsilon}(t) \end{aligned} \tag{4.25}$$

The standard deviation of the input noise $\boldsymbol{\omega}$ is set to create a signal-to-noise ratio of the input $\boldsymbol{u}_\alpha$ of 26 dB. The measurement noise $\boldsymbol{\upsilon}$ is set to the same signal-to-noise ratio for the output values $\boldsymbol{y}$. The sampling rate of the inputs δ_u as well as for the measurements δ_y is set to one hour. An exception is the Kalman filter, where the sampling rate of the inputs δ_u is set to one minute to achieve more accurate predictions. The prediction results are compared with the simulated values between times t_{k_1} and t_{k_2} with $k_2 > k_1$ and thus following definition (2.6): $t_{k_2} > t_{k_1}$. As the predicted results come with a sampling time δ_y of one hour but the sampling rate of the system δ is one day, the k's have to be replaced with j's as defined in eq. (4.14). Let us set $t_{j_1} = t_{k_1}$ and $t_{j_2} = t_{k_2}$ with $j_1 \overset{(4.15)}{=} N_{\delta_\mathrm{y}} \cdot k_1$ and $j_2 \overset{(4.15)}{=} N_{\delta_\mathrm{y}} \cdot k_2$. Further, the following notations for each state vector component $i_\mathrm{x} = 1, \ldots, n_\mathrm{x}$ shall be defined:

$$\begin{aligned} \boldsymbol{x}_{i_\mathrm{x},[j_1,j_2]} &:= \left[x_{i_\mathrm{x}}\left(t_{j_1}\right), x_{i_\mathrm{x}}\left(t_{j_1} + \delta_\mathrm{y}\right), \ldots, x_{i_\mathrm{x}}\left(t_{j_2}\right)\right]^T \\ &\overset{(4.14)}{=} \left[x_{i_\mathrm{x}}\left(t_{j_1}\right), x_{i_\mathrm{x}}\left(t_{j_1+1}\right), \ldots, x_{i_\mathrm{x}}\left(t_{j_2}\right)\right]^T \\ &\overset{(4.16)}{=} \left[x_{i_\mathrm{x},j_1}, x_{i_\mathrm{x},j_1+1}, \ldots, x_{i_\mathrm{x},j_2}\right]^T \end{aligned} \tag{4.26}$$

and

$$\begin{aligned} \boldsymbol{x}_{i_\mathrm{x},[k_1,k_2]} &:= \left[x_{i_\mathrm{x}}\left(t_{k_1}\right), x_{i_\mathrm{x}}\left(t_{k_1} + \delta\right), \ldots, x_{i_\mathrm{x}}\left(t_{k_2}\right)\right]^T \\ &\overset{(2.6)}{=} \left[x_{i_\mathrm{x}}\left(t_{k_1}\right), x_{i_\mathrm{x}}\left(t_{k_1+1}\right), \ldots, x_{i_\mathrm{x}}\left(t_{k_2}\right)\right]^T \\ &\overset{(4.17)}{=} \left[x_{i_\mathrm{x},k_1}, x_{i_\mathrm{x},k_1+1}, \ldots, x_{i_\mathrm{x},k_2}\right]^T. \end{aligned} \tag{4.27}$$

To do a fair comparison between all three methods, the estimated and simulated state vector components are compared starting at day $k_1 = 31$ until day $k_2 = 100$. The reason is, that the soft sensor-based method returns its earliest state estimate at day 31 (see Section 4.4.2) whereas the other two methods start at time 0. As $N_{\delta_\mathrm{y}} \overset{(4.2)}{=} \frac{\delta}{\delta_\mathrm{y}} = 24$ it is $j_1 \overset{(4.15)}{=} 24 \cdot 31$ and $j_2 \overset{(4.15)}{=} 24 \cdot 100$. As performance measure for the i_xth estimated state vector component $\hat{x}_{i_\mathrm{x}}$ the root-mean-square error (RMSE) $e_{\hat{\boldsymbol{x}},i_\mathrm{x}} \in \mathbb{R}^+$ is used:

$$e_{\hat{\boldsymbol{x}},i_\mathrm{x}} := \sqrt{\frac{1}{j_2 - j_1 + 1}\left(\hat{\boldsymbol{x}}_{i_\mathrm{x},[j_1,j_2]} - \boldsymbol{x}_{i_\mathrm{x},[j_1,j_2]}\right)^T \cdot \left(\hat{\boldsymbol{x}}_{i_\mathrm{x},[j_1,j_2]} - \boldsymbol{x}_{i_\mathrm{x},[j_1,j_2]}\right)} \tag{4.28}$$

and the total performance measure $e_{\hat{\boldsymbol{x}}} \in \mathbb{R}^+$ is the euclidian mean value of all n_x RMSE

values $e_{\hat{\boldsymbol{x}},i_\mathrm{x}}$:

$$e_{\hat{\boldsymbol{x}}} := \frac{1}{n_\mathrm{x}} \cdot \sum_{i_\mathrm{x}=1}^{n_\mathrm{x}} e_{\hat{\boldsymbol{x}},i_\mathrm{x}} \tag{4.29}$$

4.4.2 State Estimation using Software Sensors

The state estimator is configured as follows. As classification method Random Forests is used with 50 trees and $C = 50$ classes. The training and validation data are generated out of totally 60.000 simulated days. $N_\mathrm{u} = 5$ input data filters are used, their window sizes are set to $\mathcal{W}_\mathrm{u} = \{12, 24, 3 \cdot 24, 7 \cdot 24, 14 \cdot 24\}$. Remember, that these window sizes $w_{\mathrm{u},i_{\boldsymbol{\Lambda}_\mathrm{u}}}$ are measured in the sampling time of the inputs δ_u, see eq. (4.6). As output filters the following $N_\mathrm{y} = 7$ are used: $\mathcal{W}_\mathrm{y} = \{12, 24, 3 \cdot 24, 7 \cdot 24, 14 \cdot 24, 21 \cdot 24, 31 \cdot 24\}$, eq. (4.8). In Figure 4.3 an example result for the simulation starting at the eighth initial state with the input trajectory $\boldsymbol{u}_3$ is shown. It can be seen, that the simulated

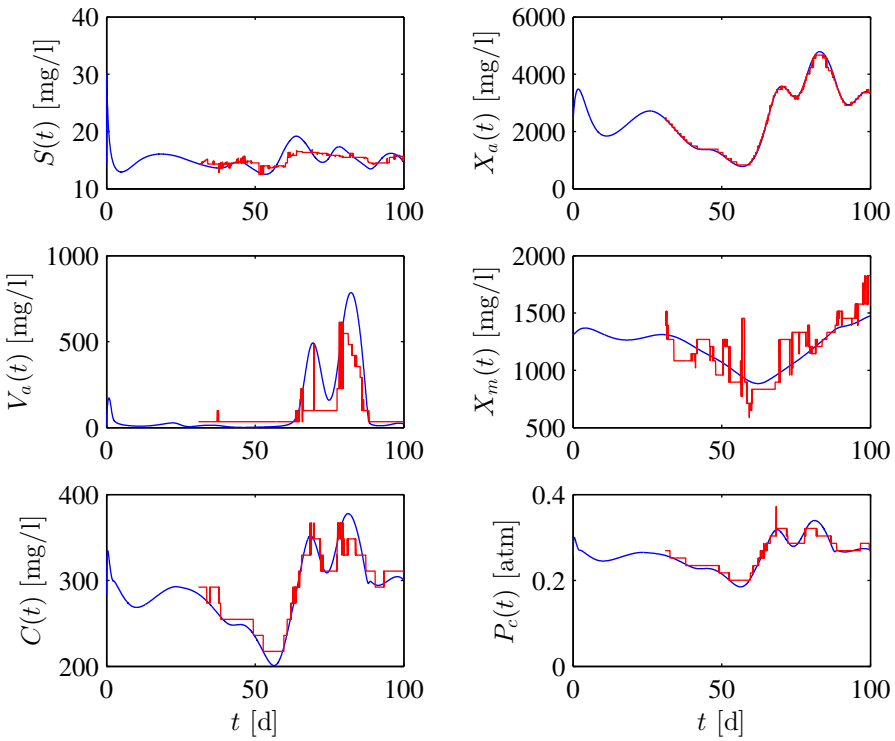

Figure 4.3: Results for state estimation using software sensors for the experiment starting at the eighth initial state with the input trajectory $\boldsymbol{u}_3$. The simulation results are coloured in blue and the estimation results in red.

states for some components are estimated quite accurately (e.g. X_a), but others show

a considerable estimation error (e.g. X_m). Nevertheless, in principle, the states are predicted with an almost constant accuracy for all 30 performed experiments as can be seen in the left box plot in Figure 4.6. There, the error measure defined in eq. (4.29) is shown.

4.4.3 Hybrid Extended Kalman Filter

The hybrid extended Kalman filter is initialized as described in Section 4.4.3. Furthermore, the covariance matrix of the process noise $\boldsymbol{\Psi}_{\boldsymbol{\omega}}$ is set to

$$\boldsymbol{\Psi}_{\boldsymbol{\omega}} = \begin{pmatrix} {\sigma_{\omega_1}}^2 & 0 \\ 0 & {\sigma_{\omega_2}}^2 \end{pmatrix}$$

and the covariance matrix of the measurement noise $\boldsymbol{\Psi}_{\boldsymbol{\upsilon}}$ to

$$\boldsymbol{\Psi}_{\boldsymbol{\upsilon}} = \begin{pmatrix} {\sigma_{\upsilon_1}}^2 & 0 \\ 0 & {\sigma_{\upsilon_2}}^2 \end{pmatrix}.$$

There, $\sigma_{\omega_{i_\mathrm{u}}} \in \mathbb{R}^+$ is the standard deviation of the process noise ω_{i_u} added to input $i_\mathrm{u} = 1, 2$ and $\sigma_{\upsilon_{i_\mathrm{y}}} \in \mathbb{R}^+$ is the standard deviation of the measurement noise υ_{i_y} added to output $i_\mathrm{y} = 1, 2$. The initial state of the model $\boldsymbol{x}_0$ is estimated with three different accuracies: 1 %, 5 % and 10 %, modelled as normal distributed noise. So, in total 90 simulations are performed, 30 for each initial state estimation accuracy. In Figure 4.4 the results of one experiment are shown. As can be seen, the simulated values are estimated with a very high accuracy. Nevertheless, the disadvantage of the Kalman filter is its dependency of an accurate estimate of the initial state (Reif et al., 1999, 2000). This can be seen in Figure 4.6, where the results in dependency of the accuracy of the initial state estimate are shown.

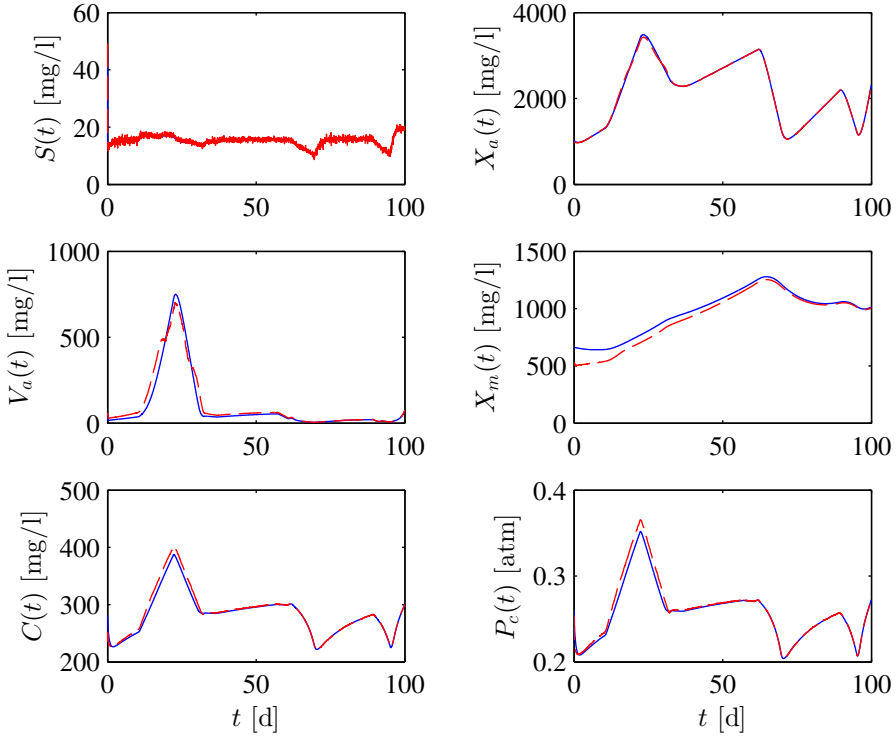

Figure 4.4: Results for state estimation using hybrid EKF. Experiment starting at fourth state, with $\boldsymbol{u}_2$ as input and 10 % as uncertainty in initial state estimate $\hat{\boldsymbol{x}}_0^+$. The simulation results are coloured in blue (solid) and the estimation results in red (dashed).

4.4.4 Moving Horizon Estimation

The initial state estimate at the beginning of the 90 simulations $\tilde{\boldsymbol{x}}(0)$ is known with the same uncertainty as in the case of the Kalman filter, see Section 4.4.3. With the sampling rate δ equal to one day, the window between two iterations is automatically shifted for one day (see eq. (4.23)). The length of the moving horizon $\tilde{w}_{\mathrm{MHE}}$ is set to 31, thus equal to the largest window length of the output filters in the software sensor approach, see Section 4.4.2. The optimization problem defined in eq. (4.23) is solved using CMA-ES (Hansen, 2006) with a population size of 20 and four iterations. The lower $\boldsymbol{x}_{\mathrm{LB}}$ and upper bounds $\boldsymbol{x}_{\mathrm{UB}}$ for the optimization variable $\boldsymbol{x}_{\circ}$ are set to $\tilde{\boldsymbol{x}}(t_k - \delta_{\mathrm{MHE}}) \cdot (1 \pm 0.15)$. In Figure 4.5 the result for one of the estimation experiments is shown. The estimation results are very accurate, but not as accurate as the ones for

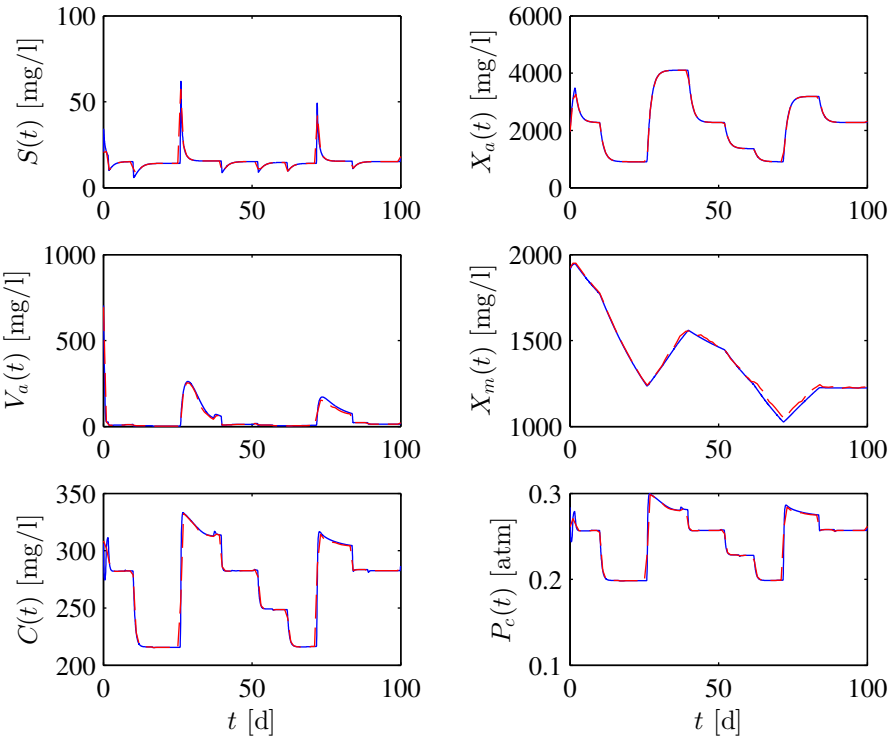

Figure 4.5: Results for state estimation using moving horizon estimation. Experiment starting at first state, with $\boldsymbol{u}_1$ as input and 1 % as uncertainty in initial state estimate $\tilde{\boldsymbol{x}}(0)$. The simulation results are coloured in blue (solid) and the estimation results in red (dashed).

the extended Kalman filter (at least for the configuration used here). Nevertheless, the clear advantage of this approach in comparison to the Kalman filter is its robustness against poor estimates of the initial state $\tilde{\boldsymbol{x}}(0)$ as can be seen in Figure 4.6, see also

(Haseltine and Rawlings, 2005). It could surely be more robust, for example when the arrival cost in the optimization problem (4.23) is weighted with the inverse covariance matrix of the state estimate as is e.g. done in Busch et al. (2013).

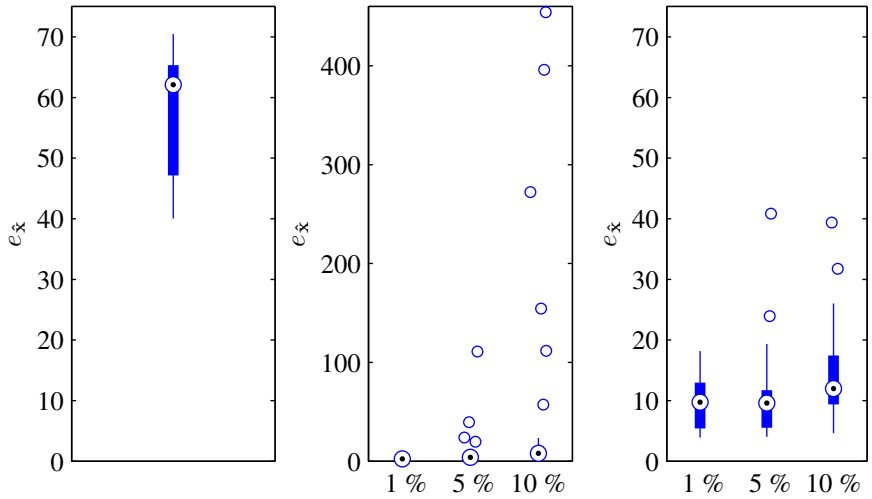

Figure 4.6: Box plots showing the error measure defined in eq. (4.29) evaluated for all 90 (30 for the first approach) simulations for the three methods. Left: Results for the soft sensor state estimation approach, Section 4.4.2. Middle: Dependency of EKF state estimation results on uncertainty in initial state estimate $\hat{\boldsymbol{x}}_0^+$ with 1 %, 5 % and 10 % normally distributed noise. Right: Results for moving horizon estimation with dependency on uncertainty in initial state estimate as well.

4.5 Summary and Discussion

In this chapter a soft sensor based state estimation method is developed. At hand of a simple anaerobic digestion model it is compared with the hybrid extended Kalman filter and moving horizon estimation.

The main advantage of the soft sensor based state estimation approach compared with conventional state estimation filters (Rawlings and Bakshi, 2006) is that an initial guess of the initial state of the system is not necessary. Furthermore, the state estimator is a static function, such that stability issues and drift are not existent. However, the yield estimation accuracy cannot compete with the approaches extended Kalman filter and moving horizon estimation. Furthermore, the first state estimate in this configuration arrives not before the 31st day of the simulation (see Section 4.4.2). On a real plant this means that measurement data of the last 31 days must be available before this estimator can be used. Before that the plant must be operated by hand. Overall, the hybrid extended Kalman filter yields the best, but also the worst estimation results in dependence of the accuracy of the initial state estimate $\hat{\boldsymbol{x}}_0^+$, see Figure 4.6. This and

the problem that the linearized model (eq. (4.19)) is required to be locally observable (Dewil et al., 2011) makes it almost impossible to use for larger anaerobic digestion models such as the ADM1. The reason is, that on biogas plants in practice only a few process values can be measured, such that the plant is not observable.
In conclusion the moving horizon estimation approach is seen as the best tested state estimation candidate for anaerobic digestion processes. Among its advantages are its robustness and explicit incorporation of state constraints and parameter estimation schemes (Rao, 2000, Rao et al., 2003). One challenge of MHE is to implement it for real-time application. The MATLAB® implementation on a standard computer[1] needs for one sample time δ a mean runtime of 1.55 s. In comparison the mean runtime of the hybrid extended Kalman filter is 0.18 s and for the soft sensor approach it is even only 0.03 s. But, these results should only be seen as rough reference values because the implementations were not written for high performance.
As the to be solved optimization problem in MHE (eq. (4.23)) is quite complex and time-consuming in this thesis this approach was not implemented for the ADM1. But, the reader is highly encouraged to spend the effort and to apply moving horizon estimation to the ADM1. How robust the moving horizon estimation approach is against the non-observability of the ADM1 in practice would be interesting to see.
To summarize, in this thesis the self-developed soft sensor based approach is used for state estimation of the ADM1. Results for that can be found in Chapter 8 and its use in a closed-loop control in Chapter 9.

[1] Intel® Core™ i7-2600 CPU @ 3.40 GHz, 8.00 GB RAM, Windows 8, 64 bit